Playing with Parameters: Cross-parameterization in Graphs

Nicolas Bourgeois, Konrad Kazimierz Dabrowski, Marc Demange, Vangelis Paschos

Laboratoire d'Analyses et Modélisation de Systèmes pour l'Aide à la Décision
UMR 7243

CAHIER DU LAMSADE

342

Septembre 2013

Playing with parameters: Cross-parameterization in Graphs

N. Bourgeois, K.K. Dabrowski, M. Demange, V.Th. Paschos

Playing with Parameters: Cross-parameterization in Graphs*

N. Bourgeois[(a)] K.K. Dabrowski[(b)] M. Demange[(b)]

V. Th. Paschos[(c)]

September 20, 2013

Abstract

When considering a graph problem from a parameterized point of view, the parameter chosen is often the size of an optimal solution of this problem (the "standard"). A natural subject for investigation is what happens when we parameterize such a problem by the size of an optimal solution of a different problem. We provide a framework for doing such analysis. In particular, we investigate seven natural vertex problems, along with their respective parameters: α (the size of a maximum independent set), τ (the size of a minimum vertex cover), ω (the size of a maximum clique), χ (the chromatic number), γ (the size of a minimum dominating set), i (the size of a minimum independent dominating set) and ν (the size of a minimum feedback vertex set). We study the parameterized complexity of each of these problems with respect to the standard parameter of the others.

1 Introduction: cross-parameterization

Parameterized complexity has been widely studied over the past few years. The main motivation for this area is to study the tractability of a problem with respect to the size of some of its parameters besides the size of the instance. In particular, some NP-hard problems may become more tractable for instances with small parameter value (see the books [5, 8, 12] for more details about parameterized complexity). From a practical point of view this may be interesting when the considered parameter has a strong dependence on the underlying model, in which case instances with low parameter value are relevant. Most

*This work was supported the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010

[(a)]Université Paris 1, SAMM, nbourgeo@phare.normalesup.org

[(b)]ESSEC Business School, {dabrowski,demange}@essec.edu

[(c)]PSL Research University, Université Paris-Dauphine, LAMSADE CNRS, UMR 724 and Institut Universitaire de France, paschos@lamsade.dauphine.fr

of the time in the related literature, the main parameter used for an optimization problem is the optimal value itself. This causes two limitations: first, it sometimes becomes difficult to compare the parameterized complexity of two different problems, each of them dealing with a specific parameter. Second, it may happen that instances with small parameter value but large optimal value are relevant and in this case an approach with this specific parameter allows us to solve such instances efficiently.

In this paper we propose a framework for considering the complexity of any problem with respect to any kind of parameter. Considering the parameterized complexity of a problem with respect to several parameter gives more information and deeper insight to the real tractability of a problem. It also provides a more stable framework for comparing the tractability of different problems. If we consider standard parameters for a series of problems, then it could be relevant to study parameterized complexity for each of the problems considered with respect to the standard parameters of the others. This is what we call *cross-parameterization*, in what follows. Here, we draw a first framework for addressing this question and we test its relevance by considering seven very natural combinatorial optimization problems on graphs, leading to seven related parameters. Our aim here is to study the tractability of each problem under each considered parameter (see Table 1). However, this can be done for any other combinatorial problem and any parameter.

When handling optimization problems, three different versions of a problem can be considered: either computing, for any instance, an optimal solution (the *constructive* version), calculating the optimal value (the *non-constructive* case) or solving the decision version of a problem. We include this distinction in our general framework (see Section 2), defining a general notion of FPT reducibility and proving the equivalence between these different approaches for a wide range of problems including our seven basic graph problems. Then, in Section 3, we prove the results of Table 1, considering first tractability and then intractability results. Some of the proofs in this section rely on results given in Section 2.

1.1 Notation

As a first attempt to draw such a general cross-parameterization framework, we have selected seven basic vertex parameters in graphs:

- α: the size of a maximum independent set;
- τ: the size of a minimum vertex cover;
- ω: the size of a maximum clique;
- χ: the chromatic number;
- γ: the size of a minimum dominating set;
- i: the size of a minimum independent dominating set;
- ν: the size of a minimum feedback vertex set.

We also write n for the number of vertices and Δ for the maximum degree. All these values are integer graph-parameters and we will use p to refer to any general graph parameter, i.e. p is any computable function that takes a graph as input and outputs an integer value. Here we will mainly consider parameters with non-negative integral values. For a graph G, $p(G)$ denotes the value of the related parameter for G.

For any such parameter p, there is an associated combinatorial problem Π^p, which is that of computing the parameter for a given graph. Here, the considered parameters are in fact the value of the optimization problem Π^p and in Section 2 we will distinguish between computing the parameter itself and computing a corresponding optimal solution. A graph problem asking for a value, solution or answer to a decision question is said to be *fixed-parameter tractable* (FPT) with respect to a parameter p (or simply FPT(p)) if there is an algorithm that solves the problem in $O(g(p(G))P(n(G))$ time on input graph G, where g is a computable function and P is a polynomial function. Without loss of generality we assume that g is non-decreasing (otherwise replace $g(p)$ by $\max_{p' \leqslant p} g(p')$). When no ambiguity occurs, we write $O(f(p)P(n)) = O^*(f(p))$. We say that such an algorithm is an FPT algorithm and runs in FPT time. We will sometimes write (Π, p) to refer to the problem Π parameterized by p.

For a graph G with a vertex x, the set $N(x)$ denotes the neighbourhood of x, i.e. the set of vertices adjacent to x. We set $N[x] = N(x) \cup \{x\}$, the closed neighbourhood of x. If D is a set of vertices, $N[D]$ denotes the union of the closed neighbourhoods of vertices in D. Finally, $V(G)$ denotes the vertex set of G and for $V' \subset V(G)$, $G[V']$ denotes the sub-graph of G induced by V'.

1.2 Our results

To date, research has mostly focused on the complexity of a problem when parameterized by the solution size, so we can already fill the diagonal line of Table 1. Some problems have been shown to be fixed parameter-tractable: Π^τ and Π^α are FPT(τ) [1], and Π^ν is FPT(ν) [7].

On the other hand, Π^ω (resp. Π^α) is a classic example of a problem which is W[1](ω)-complete (resp. W[1](α)-complete) [6], while Π^γ and Π^i are W[2](γ)-complete and W[2](i)-complete, respectively [4]. Problem Π^χ is $\notin$XP(χ) since chromatic number remains NP-hard when the optimum is 3 [11, 14]. Here, we take $\notin$XP to mean that the problem is not in the class XP (assuming P $\neq$ NP). Note that a any vertex cover is consists of vertices whose removal leaves an independent set (and vice versa). Therefore, any minimum vertex cover consists of precisely those vertices that are not in some maximum independent set. Therefore, any results for the complexity of Π^α will also apply to the complexity of Π^τ and vice versa(as suggested in Table 1).

The overall results are summarized in Table 1 where, for a complexity class C, C-c (resp. C-h) means C-complete (resp. C-hard).

	Π^ω	Π^χ	Π^γ	Π^i	Π^ν	Π^α/Π^τ
ω	W[1]-c	$\notin$XP	$\notin$XP	$\notin$XP	$\notin$XP	$\notin$XP
χ	W[1]-h	$\notin$XP	$\notin$XP	$\notin$XP	$\notin$XP	$\notin$XP
γ	$\notin$XP	$\notin$XP	W[2]-c	$\notin$XP	$\notin$XP	$\notin$XP
i	$\notin$XP	$\notin$XP	W[2]-h	W[2]-c	$\notin$XP	$\notin$XP
ν	FPT	FPT	FPT	FPT	FPT	FPT
τ	FPT	FPT	FPT	FPT	FPT	FPT
α	$\notin$XP	$\notin$XP	W[2]-h	W[2]-h	W[1]-h	W[1]-c

Table 1: A summary of our results. The columns represent graph problems. The rows represent parameters.

2 Constructive vs. non-constructive computation

Before going into the proofs of the main results reported in Table 1, we first revisit, in the context of the present framework, the question of equivalence between computing an optimal solution of an optimization problem, computing its optimal value and the related decision problem. In particular, we will make use of this distinction in the proof of Proposition 11.

Definition 1. *Any instance of an optimization problem* Π *can be expressed as a mathematical program of the form below with objective function* f *and constraint set* $\mathcal{C}$*:*

$$\begin{cases} \max \text{ or } \min f(x) \\ x \in \mathcal{C} \end{cases} \tag{1}$$

When dealing with such optimization problem several frameworks can be considered, leading to three different versions of the problem. The *constructive* version Π_c asks us to compute an *optimal solution* for the input instance, while the *non-constructive* (or *value*) version Π_v only asks us to compute an optimal value. Finally, the decision version Π_d asks us to decide, for any value k, whether there is some feasible solution $x \in \mathcal{C}$ satisfying $f(x) \geqslant k$ (if Π is a maximization problem) or $f(x) \leqslant k$ (if Π is a minimization problem). When needed, we will denote any problem with its version Π_t, with $t \in \{c, v, d\}$; when this subscript is not specified, we will consider the constructive version $t = c$.

This distinction, along with the relative complexity of the different versions has been considered several times in the literature, in particular for the classical complexity framework [13] and for the framework of polynomial approximation [3]. This same distinction can be considered in the frame of parameterized complexity. Most often, negative results are stated for the value-version while positive results are stated for the constructive version. Note that the value version is not more difficult than the constructive one as long as the objective function can be computed in reasonable time. More precisely if f can be computed in polynomial time (resp. FPT time) then any polynomial (resp. FPT) algorithm for the constructive version can immediately be turned into a polynomial (resp. FPT) algorithm for the value version. The same holds between

the value version and the decision version, the former being at least as difficult as the latter if f can be efficiently computed.

In this paper we only consider problems for which f can be computed in polynomial time and consequently the constructive version is at least as hard as the non-constructive one, which itself is also as hard as the decision version. An interesting question is whether or not these versions are equivalent in complexity. To study the relative complexity of problems and, in particular of the different versions of a problem, the notion of *reduction* is useful. Many kinds of reductions (mainly polynomial ones) have been introduced in the literature, allowing us to compare tractability of different problems and even between the different versions of a given problem. The notion of *FPT reduction* able to transfer FPT algorithms from one problem to another one has also been introduced (see e.g. [5, 8, 12]). Here, we somewhat enhance it in order to integrate the possibility of considering any kind of parameters for the considered problems.

Definition 2. *Let $(\Pi^1_{t_1}, p_1), (\Pi^2_{t_2}, p_2)$ be two optimization problems parameterized by p_1 and p_2 respectively, with $t_1, t_2 \in \{c, v, d\}$. An FPT-reduction from $(\Pi^1_{t_1}, p_1)$ to $(\Pi^2_{t_2}, p_2)$ is an algorithm solving $\mathcal{A}_1$ in FPT time with respect to parameter p_1 using an Oracle $\mathcal{O}_2$ for $\Pi^2_{t_2}$ such that for any instance I_1 of $\Pi^1_{t_1}$, any call on $\mathcal{O}_2$ is made on instances I'_2 for $\Pi^2_{t_2}$ whose size is polynomially bounded with respect to I_1 and which satisfy $p_2(I'_2) \leqslant h(p_1(I_1))$ for some function h. We then say that $(\Pi^1_{t_1}, p_1)$ FPT-reduces to $(\Pi^2_{t_2}, p_2)$ and denote this by $(\Pi^1_{t_1}, p_1) \leqslant_{FPT} (\Pi^2_{t_2}, p_2)$. If the reduction is polynomial, we denote it by $\Pi^1_{t_1} \leqslant_P \Pi^2_{t_2}$ (no parameter needs to be specified).*

Note that, since $\mathcal{A}_1$ is FPT, the number of calls to Oracle $\mathcal{O}_2$ is bounded by an FPT function with respect to parameter p_1 and consequently if $\mathcal{O}_2$ is an FPT algorithm with respect to p_2, then the conditions on $p_2(I'_2)$ and on the size of I'_2 ensures that the reduction leads to an FPT algorithm for $\Pi^1_{t_1}$ with respect to parameter p_1. In other words, such a reduction is able to transform an FPT algorithm for $\Pi^2_{t_2}$ with respect to parameter p_2 into an FPT algorithm for $\Pi^1_{t_1}$ with respect to parameter p_1.

Note that the decision version and the non-constructive version are equivalent for a very large class of problems, as stated by the following proposition:

Proposition 1. *Let Π be an optimization problem, an instance of which is defined as in Definition 1, with parameter p satisfying:*

1. *f has integral values;*
2. *the output value associated to some feasible input can be found in polynomial time;*
3. *there is a polynomial function P and a function ℓ such that $\forall x, y \in \mathcal{C}, |f(x) - f(y)| \leqslant 2^{\ell(p)P(n)}$.*

Then $(\Pi_v, p) \leqslant_{FPT} (\Pi_d, p)$.

In particular, this holds for integral non-negative objective functions bounded by $2^{\ell(p)P(n)}$.

Proof. The reduction is easily done by binary search. Without loss of generality we assume that Π is a maximization problem (the minimization case is similar). We start by finding a feasible output value. Next, we try to find a $K \leqslant 2^{\ell(p)P(n)+1}$ such that the optimal value lies in $[\lambda, \lambda + K]$. Note that $\ell(p)$ may not be explicitly known. We ask the Oracle $\mathcal{O}$ for the problem Π_d, whether $\exists x \in \mathcal{C}, f(x) \geqslant \lambda + 2^k$ for successive values of $k \geqslant 1$, or not. Let K be the first value for which the answer is NO. We know that $K \leqslant 2^{\ell(p)P(n)+1}$ by hypothesis. We then find the optimal value by binary search in the interval $[\lambda, \lambda + K]$, using Oracle $\mathcal{O}$.

This process is FPT with respect to parameter p since the number of calls to Oracle $\mathcal{O}$ is at most $2\ell(p)P(n) + 4$, each time for the same Π-instance. □

For any optimization problem Π and parameter p, we let $\Pi|_{p-\text{bounded}}$ denote the sub-problem of Π restricted to instances where $p \leqslant K$ for a fixed bound K. In [13] a general process was proposed to reduce the constructive version Π_c of an optimization problem Π to its non-constructive version Π_v. The main idea is to transform Π into Π' by transforming the objective function f into a one-to-one function f' such that $\forall x, y \in \mathcal{C}, f(x) \leqslant f(y) \Rightarrow f'(x) \leqslant f'(y)$.

Moreover, one needs to suppose that the inverse function f'^{-1} can be computed in polynomial time. In particular, this holds when f has integral values, there is a polynomial-time computable bound B such that $|\mathcal{C}| \leqslant B$ and there is a total order on $\mathcal{C}$ such that the related rank-function r as well as its inverse function r^{-1} are both polynomially computable. Typically, for $\mathcal{C} \subset \{0,1\}^{P(n)}$, where P is a polynomial function, the lexicographic order can be computed and inverted in polynomial time.

We can then take $f'(x) = (B+1)f(x) + r(x)$. Note that if such a function f' does exist and if $\Pi_d \in NP$, then $\Pi'_d \in NP$.

Proposition 2. *Suppose $\Pi_d \in NP$, $(\Pi|_{p-\text{bounded}})_d$ is NP-complete for a parameter p and there is a polynomial function P such that any instance of Π satisfies $\mathcal{C} \subseteq \{0,1\}^{P(n)}$. Then $(\Pi_c, p) \leqslant_{FPT} (\Pi_v, p)$.*

Proof. We have $\Pi_c \leqslant_P \Pi'_c$ since both problems have exactly the same feasible solutions and any optimal solution to Π'_c is also optimal for Π_c. $\Pi'_c \leqslant_P \Pi'_v$ by definition of Π' (see above).

Taking the bound $B = 2^{P(n)}$ we get $\Pi'_v \leqslant_P \Pi'_d$ and since Π'_d is in NP and $(\Pi|_{p-\text{bounded}})_d$ is NP-complete, $\Pi'_d \leqslant_P (\Pi|_{p-\text{bounded}})_d$. Given an instance of (Π_c, p), the reduction (see Definition 2) simply computes an equivalent instance of $((\Pi|_{p-\text{bounded}})_d, p)$, where $p \leqslant K$, for a constant K, h is the constant function equal to K and $\mathcal{O}$ is an oracle for $(\Pi|_{p-\text{bounded}})_d$. Note that any FPT algorithm for (Π, p) leads to a polynomial-time algorithm for $(\Pi|_{p-\text{bounded}})_d$, and consequently such a reduction transforms an FPT algorithm into a polynomial time one. □

Proposition 3. *We have $(\Pi_c, p) \leqslant_{FPT} (\Pi_v, p)$ for:*

1. $\Pi^\alpha, \Pi^\chi, \Pi^\gamma, \Pi^i, \Pi^\nu$ *and* $p \in \{\omega, \chi\}$;

2. Π^{ω}, Π^{χ} *and* $p = \alpha$*;*

3. Π^{i} *and* $p = \gamma$*;*

4. $\Pi^{\alpha}, \Pi^{\tau}, \Pi^{\omega}, \Pi^{\chi}, \Pi^{\nu}$ *and* $p \in \{\gamma, i\}$.

Proof. All these results are the consequence of Proposition 2. For all these problems $\mathcal{C} \subseteq \{0,1\}^{P(n)}$ for some polynomial function P and the decision version is known to be in NP. So, we need to show that in each case the problem $(\Pi|_{p-\text{bounded}})_d$ is NP-complete.

1. This follows from the inequality $\Delta(G) + 1 \geqslant \chi(G) \geqslant \omega(G)$. We simply recall that all these problems remain NP-complete on graphs of degree 3 [9], and that 3-colouring is NP-complete [9].

2. Using the previous results in $\overline{G}$ we get the NP-completeness of Π^{ω}_d in graphs whose maximum independent set is of size 3. A colouring in G induces a clique partition in $\overline{G}$. The decision version of MINIMUM CLIQUE PARTITION is NP-complete in graphs of maximum degree 3 [2], and thus in graphs of maximum clique 4. Thus, Π^{χ}_d is NP-complete in graphs whose maximum independent set is of size 4.

3. Let $G = (V, E)$ be an arbitrary non-empty graph. Let G_0 and G_1 be disjoint copies of G. We form the graph G' by taking the disjoint union of G_0 and G_1, adding a vertex u which dominates the vertices of G_0, adding a vertex v which dominates G_1 and joining u to v with an edge. Then $\gamma(G') = 2$ (since $\{u, v\}$ is a dominating set and there is no dominating vertex). Consider an independent dominating set I in G'. It cannot contain both u and v. If it contains neither, then $|I| \geqslant 2i(G) \geqslant i(G) + 1$. If it contains exactly one of u and v (without loss of generality v), then it must contain an independent dominating set for G_1, so $|I| \geqslant i(G) + 1$. But any independent dominating set of G_1, together with u is an independent dominating set of G'. Thus $i(G') = i(G) + 1$, so Π^{i}_d is NP-complete, even for graphs with $\gamma = 2$.

4. Suppose $G = (V, E)$ is a non-empty graph. Let $\widetilde{G}$ be the graph obtained from G by adding a new vertex v adjacent to all of V. Then we have the following:

- $\alpha(\widetilde{G}) = \alpha(G)$;
- $\omega(\widetilde{G}) = \omega(G) + 1$;
- $\chi(\widetilde{G}) = \chi(G) + 1$;
- $\tau(\widetilde{G}) = \tau(G) + 1$;
- $\nu(\widetilde{G}) = \nu(G) + 1$.

The first three items are obvious. The fourth follows from the fact $\alpha + \tau = n$. For the last one, it is clear that $\nu(\widetilde{G}) \leqslant \nu(G) + 1$, since adding the dominating vertex to any feedback vertex set of G yields a feedback vertex set of $\widetilde{G}$. Consider a minimal feedback vertex set F of $\widetilde{G}$. Let v be the dominating vertex of $\widetilde{G}$. We

want to show that $|F| \geqslant \nu(G) + 1$. If $v \in F$, then $F \setminus \{v\}$ is a feedback vertex set of G, and we are done. Suppose $v \notin F$. Then $G \setminus F$ must be a stable set (otherwise two adjacent vertices in G, together with v would form a C_3 in $\widetilde{G}$). Let w be an arbitrary vertex in F. Then $F \setminus \{w\}$ must be a feedback vertex set in G, since $N_{G\setminus(F\setminus\{w\})}(w)$ is an independent set. Thus $|F| \geqslant \nu(G) + 1$ for all minimal feedback vertex sets of $\widetilde{G}$, i.e. $\widetilde{G} \geqslant \nu(G) + 1$.

Note also that $\gamma(\widetilde{G}) = i(\widetilde{G}) = 1$ and consequently the decision version of Π^α, Π^τ, Π^ω, Π^χ and Π^ν all remain NP-complete in graphs with $\gamma = i = 1$. This completes the proof. □

Another case where equivalence between constructive and non-constructive cases can be easily stated is the hereditary case. A property $h : 2^V \Longrightarrow \{0, 1\}$ for a finite set V is *hereditary* if $h(U') \geqslant h(U), \forall U' \subset U \subset V$. We then consider an *hereditary maximization* problem, an instance of which can be written

$$\begin{cases} \max f(U) \\ h(U) = 1, U \subseteq 2^V \end{cases} \tag{2}$$

The size of this instance is $|V|$ and any subset $V' \subset V$ also defines an instance of Π.

Proposition 4. *Let* Π *be an hereditary maximization problem and consider an* non-decreasing *parameter* p*, meaning that* $U' \subset U \Rightarrow p(U') \leqslant p(U)$*. Then* $(\Pi_c, p) \leqslant_{FPT} (\Pi_v, p)$.

Proof. Considering an instance V of Π_c and let $S = \emptyset$. The reduction consists of taking any element $v \in V$ and testing the value of the instance $V \setminus \{v\}$. If it is smaller than the value of the instance V, then every optimal solution must include v, in which case we add it to a set S. If after removing v, the optimal value is unchanged, there must be an optimal solution which does not contain v, in which case we delete v from the graph. We continue this process, deleting vertices from the graph where possible until all remaining vertices belong to S. Then S is an optimal solution. In all, we will need $O(|V|)$ requests to an Oracle $\mathcal{O}$ for Π_v on sub-instances of V. The hypothesis on the parameter makes the whole process FPT with respect to p if the oracle is also FPT. This concludes the proof. □

Corollary 1. *It holds that* $(\Pi_c, p) \leqslant_{FPT} (\Pi_v, p)$ *for* $\Pi \in \{\Pi^\alpha, \Pi^\omega, \Pi^\nu, \Pi^\tau\}$ *and* $p \in \{\alpha, \omega, \chi, \nu, \tau\}$.

Note also that for any constructive problem that can be solved in FPT time with respect to a given parameter p, the FPT equivalence between non-constructive and constructive versions (Π_c, p) and (Π_v, p) is trivial and consequently, this holds for all problems considered in this paper under the parameters τ and ν.

Proposition 5. $(\Pi_c, p) \leqslant_{FPT} (\Pi_v, p)$ *for* $\Pi = \Pi^\gamma$*,* $p \in \{\alpha, \gamma\}$ *and* $\Pi = \Pi^i$*,* $p \in \{\alpha, i\}$.

Proof. Here we need explicit reductions. Consider a graph $G = (V, E)$ and a vertex $v \in V$. We consider the graph G' obtained from G by adding a vertex v' connected to v. We then have $\gamma(G') = \gamma(G)$ if and only if there is a minimum dominating set in G containing v. Indeed, suppose $\gamma(G') = \gamma(G)$ and consider a minimum dominating set DS' of G'. If $v' \in DS'$, then $DS' \setminus \{v'\} \cup \{v\}$ is a minimum dominating set of G' and a dominating set of G containing v. Since it is of value $\gamma(G)$, it is minimum in G. Conversely, suppose that there is a minimum dominating set DS of G containing v. It is a dominating set in G' and moreover $\gamma(G') \geqslant \gamma(G)$ since $N[v'] \subseteq N[v]$. So, an oracle for γ_v allows us to decide whether there is a minimum dominating set containing v. Since $\alpha(G') \leqslant \alpha(G) + 1$ and $\gamma(G') \leqslant \gamma(G) + 1$, an FPT oracle with respect to α or γ remains FPT with respect to the instance G when applied to G'. Using this, one can identify a first vertex v which occurs in some optimal solution of the dominating set problem. Then, for any vertex $w \neq v$, we consider the graph G'' obtained by merging v and w (i.e. replacing the vertices by a new vertex whose neighbourhood is $(N(v) \cup N(w)) \setminus \{v, w\}$). Using similar arguments, there is a minimum dominating set of G containing both v and w if and only if $\gamma(G') = \gamma(G'') + 1$. Note also that $\alpha(G'') \leqslant \alpha(G')$ and $\gamma(G'') \leqslant \gamma(G')$. By iterating this second reduction, we can identify an optimal solution. By construction, this will take $\gamma(G) - 1$ such iterated reductions.

For Π^i we devise the following reduction for deciding whether there is a minimum independent dominating set in G containing v: consider G''' obtained from G by removing $N[v]$. Then $i(G) = i(G''') + 1$ if and only if v belongs to some optimal solution. Indeed, suppose $i(G) = i(G''') + 1$ and take a minimum independent dominating set IDS''' of G'''. Then $IDS''' \cup \{v\}$ is an independent dominating set of G of value $i(G)$. Conversely, if an optimal solution IDS contains v, then $IDS \setminus \{v\}$ is an independent dominating set of G'''. Note moreover that $\alpha(G''') \leqslant \alpha(G)$ and that $i(G''') \leqslant i(G) + 1$ for any considered graph G'''. Here we simply need to iterate the process on the remaining graph.

Both reductions satisfy the conditions of Definition 2, which concludes the proof. □

Note that the previous reduction does not work for (Π^γ, i), which needs a different reduction.

Proposition 6. $(\Pi^\gamma_c, i) \leqslant_{FPT} (\Pi^\gamma_v, i)$.

Proof. Consider a graph $G = (V, E)$ and a set of vertices $V' \subset V$. We consider the graph $G_{V'}$ obtained from G by adding a stable set V'', $|V''| = |V'|$ and a perfect matching between V' and V''. Then, using similar arguments to those for Proposition 5, we have: $\gamma(G_{V'}) = \gamma(G)$ if and only if there is a minimum dominating set containing V'. If such a V' is known and $|V'| < \gamma(G)$, then one can find a new vertex $v \notin V'$ to include in the solution by testing the value of $\gamma(G_{V' \cup \{v\}})$. Since for all considered cases $i(G_{V'}) \leqslant |V'| + i(G) \leqslant \gamma(G) + i(G) \leqslant 2i(G)$ this provides the required reduction. □

To summarize, in this section we have shown the equivalence between constructive and non-constructive optimization for all problems and parameters

considered in Table 1.

Note that, if $(\Pi_c, p) \leqslant_{FPT} (\Pi_v, p)$, then positive (FPT) and hardness results equivalently hold for one or the other version. Consequently, in the two following sections, the version is not specified in the results. So, by Proposition 1, we can consider that FPT results hold for the constructive version and hardness results hold for the decision version.

3 Main results

3.1 Tractability results

We first recall the following inequalities, which will be useful in many of the proofs in this section.

$$\begin{aligned}
&\alpha + \tau = n \\
&\alpha \geqslant i \geqslant \gamma \\
&\Delta + 1 \geqslant \chi \geqslant \omega \\
&\tau \geqslant \nu
\end{aligned}$$

Proposition 7. *$\Pi^\nu, \Pi^\omega, \Pi^\alpha, \Pi^\tau$ and Π^χ are FPT(τ) and FPT(ν).*

Proof. Since $\tau \geqslant \nu$, we only need to prove that these problems are FPT(ν).

The problem Π^ν, it is known to be FPT(ν) [7]. For all of the remaining problems, we start by finding a minimum feedback vertex set F^* in running time $O^*(f(\nu))$.

Next, we consider the maximum clique problem Π^ω. A clique contains at most two vertices from the forest $V \setminus F^*$. For each subset $C \subset F^*$ which induces a clique, we search for two adjacent vertices v, v' which are also adjacent to every vertex of C, and add them to C. If we cannot find such a pair, we look for a single vertex adjacent to every vertex of C, and add it to C. If we cannot find such a vertex, we simply keep C. Finally, we return the largest clique constructed in this way. This algorithm has running time bounded above by $O^*(f(\nu)2^\nu)$.

Next, we consider the problem Π^α. For each subset $S \subset F^*$ which is independent, we can discard $N(S)$ and use a greedy algorithm to compute a maximum independent set on the remaining forest in polynomial time. This algorithm has running time bounded above by $O^*(f(\nu)2^\nu)$. Since $\tau = n - \alpha$, this also proves the corresponding result for Π^τ.

Finally, we show how to solve Π^χ. Notice that $\chi(F^*) \leqslant \chi(G) \leqslant \chi(F^*) + 2 \leqslant \nu + 2$. We take each value of $k = 1, \ldots, \nu + 2$ in turn and test if G has a k-colouring. To do this, we first find every k-colouring of F^* (which can be done in $O^*(k^\nu)$ time). For each such colouring, we test if it extends to a k-colouring of G. To do this, we try to find a k list-colouring of the forest $V \setminus F^*$. The list of admissible colours of a vertex $L(v)$ is those for which no neighbour $u \in N(v)$ is of that colour. This list-colouring problem can be solved in polynomial time [10].

Note that the algorithm will always find a valid k-colouring when $k = \nu+2$. This whole procedure runs in FPT time when parameterized by ν. This completes the proof. □

Suppose T_1 and T_2 are vertex-disjoint trees rooted at v_1 and v_2, respectively. Let $T_1 \leftarrow T_2$ be the tree rooted at v_1 obtained by taking the disjoint union of T_1 and T_2 and then joining v_1 to v_2 with an edge. Note that every rooted tree can be built from its vertex set using just this operation. Moreover, such a representation for a tree can easily be computed in linear time.

Proposition 8. *Π^γ and Π^i are FPT(ν) and FPT(τ).*

Proof. Again, we need only to prove that the problems are FPT(ν). We start with the Π^γ case. Consider a graph $G = (V, E)$. Since $\Pi^\nu \in$ FPT(ν), we can compute a feedback vertex set F^* with running time $O^*(f(\nu))$. Fix some subset $D \subset F^*$, that we assume to be the restriction of the minimum dominating set to F^*.

We now run a dynamic programming algorithm. Note that $G[V \setminus F^*]$ is a forest. For a rooted tree T (with root vertex v) which is a subtree of a tree in $G[V \setminus F^*]$, a set $S \subseteq F^*$ and a value $d \in \{0, 1, 2\}$, we define $A(T, S, d)$ to be the minimum size of a set $D' \in V(T)$ such that $N[D \cup D'] \cap F^* = S$, and:

- if $d = 0$, D' includes the vertex v and $D \cup D'$ dominates every vertex in T;
- if $d = 1$, D' does not include the vertex v, but $D \cup D'$ dominates every vertex in T;
- if $d = 2$, $D \cup D'$ dominates every vertex in $T \setminus \{v\}$, but does not dominate v.

If, for some choice of T, v, S, d no such set D' exists, we set $A(T, S, d) = \infty$.

Now, for each tree T in the forest $G \setminus F^*$, we choose an arbitrary root vertex v and find a decomposition of T using $\leftarrow$ operations.

Let T' be a subtree (rooted at v') of T that occurs in the decomposition. We will show how to calculate the value of $A(T', S, d)$ for every possible choice of S and d.

First, as a base case, we suppose that T' consists of only a single vertex v'. If $d = 0$, this corresponds to the case where $D' = \{v'\}$. Thus we set $A(T', S, 0) = 1$ if $S = N[D \cup \{v'\}] \cap F^*$ and $A(T', S, 0) = \infty$ otherwise. If $d = 1$, this corresponds to the case where $D' = \emptyset$ and v' has a neighbour in D. Thus if v' has a neighbour in D, we set $A(T', N[D] \cap F^*, 1) = 0$. If v' has no neighbour in D or $S \neq N[D] \cap F^*$, we set $A(T', S, 1) = \infty$. If $d = 2$, this corresponds to the case where $D' = \emptyset$ and v' does not have a neighbour in D. Thus if v' has no neighbour in D, we set $A(T', N[D] \cap F^*, 2) = 0$. If v' has a neighbour in D or $S \neq N[D] \cap F^*$, we set $A(T', S, 2) = \infty$.

Now suppose that T' (rooted at v') contains more than one vertex. Then $T' = T_1 \leftarrow T_2$ for some T_1, T_2 rooted at v_1, v_2 respectively, say. Note that

$v_1 = v'$, by definition of $\leftarrow$. We now show how to calculate $A(T', S, d)$ given the values for $A(T_1, S', d')$ and $A(T_2, S', d')$ for all possible choices of S' and d'.

If $d = 0$, this corresponds to the case where D' contains v'. Consider the restriction of D' to T_1. We must have that T_1 is dominated by $(D' \cap V(T_1)) \cup D$ and that it contains the root vertex of v_1. In other words, this restriction must correspond to the $A(T_1, S_1, 0)$ case for some S_1. In this case, any valid D' for T' must dominate all of T_1 and all of T_2. Since $v' \in D'$, we know that v_2 is dominated by v'. Now consider the restriction of D' to T_2. We must have that $(D' \cap V(T_2)) \cup D$ dominates $V(T_2) \setminus \{v_2\}$. Since v_1 is adjacent to v_2 and is present in D', vertex v_2 may or may not be present in D' and it may or may not be dominated by $(D' \cap V(T_2)) \cup D$. This corresponds to the $A(T_1, S_2, d')$ case for some S_2 and some $d' \in \{0, 1, 2\}$. We therefore set $A(T', S, 0)$ to be the minimum of $\{A(T_1, S_1, 0) + A(T_2, S_2, d') | d' \in \{0, 1, 2\}, S_1 \cup S_2 = S\}$.

If $d = 1$, this corresponds to the case where $v_1 \notin D'$, but v_1 is dominated by a member of D'. This dominating vertex must either be in $D \cup (V(T_1) \cap D')$ or it must be v_2. The restrictions of D' to T_1 and T_2 therefore correspond to $A(T_1, S_1, 1)$ and $A(T_2, S_2, d')$ for some $d' \in \{0, 1\}$ and some S_1 and S_2 or they correspond to $A(T_1, S_1, 2)$ and $A(T_2, S_2, 0)$ for some S_1 and S_2. Therefore, $A(T', S, 1)$ is the minimum of $\{A(T_1, S_1, d_1) + A(T_2, S_2, d_2) | (d_1, d_2) \in \{(1, 0), (1, 1), (2, 0)\}, S_1 \cup S_2 = S\}$.

If $d = 2$, this corresponds to the case where $T' \setminus \{v_1\}$ is dominated by $D \cup D'$, but v_1 is not dominated by $D \cup D'$. This means that neither v_1 nor v_2 are present in D'. The restriction of D' to T_1 must be such that $D \cup (D' \cap V(T_1))$ does not dominate v_1, which corresponds to the $A(T_1, S_1, 2)$ case, for some S_1. However, v_2 must be dominated by a vertex in $D \cup (D' \cap V(T_2))$. This corresponds to the $A(T_2, S_2, 1)$ case. Thus $A(T', S, 2)$ is the minimum of $\{A(T_1, S_1, 2) + A(T_2, S_2, 1) | S_1 \cup S_2 = S\}$.

Using the above recursion, we can calculate the value of $A(T', S, d)$ for every rooted tree T' in $G[V(G) \setminus F^*]$ in FPT time (with parameter ν). We label these trees $T_1, \ldots, T_k$.

Now, for $S \subseteq F^*$ and $i \leqslant k$, let $B(i, S)$ be the size of the smallest set $D' \in V(T_1) \cup \cdots \cup V(T_i)$ such that $N[D \cup D'] \cap (F^* \cup V(T_1) \cup \cdots \cup V(T_i)) = S \cup V(T_1) \cup \cdots \cup V(T_i)$. Note that $B(0, S) = 0$ if $S = N[D] \cap F^*$ and ∞ otherwise. Furthermore, for $i \geqslant 1$, $B(i, S)$ is the minimum of $\{B(i-1, S_1) + A(T_i, S_2, d) | S_1 \cup S_2 = S, d \in \{0, 1\}\}$. The minimum size of a dominating set whose intersection with F^* is D is then $|D| + B(k, F^*)$. All these calculations can be done in FPT time with parameter ν. We thus branch over all possible choices of D and then, for each such choice, find the size of the minimum dominating set whose intersection with F^* is D.

The argument for Π^i is similar. Again, we start by finding a minimal feedback vertex set F^*, but this time, we only consider $D \subseteq F^*$ that are independent. We define $A'(T, S, d)$ in the same way as $A(T, S, d)$, except that now we only consider sets D' such that $D \cup D'$ is independent.

We now explain how to calculate $A'(T, S, d)$ for the tree T, rooted at v. Again, we first consider the case where T contains a single vertex. If $d = 0$, this

corresponds to the case where $D' = \{v\}$. $D \cup D'$ must be independent, so v cannot have any neighbours in D. Therefore, we set $A'(T, S, 0) = 1$ if $S = N[D \cup \{v\}] \cap F^*$ and v has no neighbours in D. Otherwise, we set $A'(T', S, 0) = \infty$. If $d = 1$, this corresponds to the case where $D' = \emptyset$ and v has a neighbour in D. We therefore set $A'(T, S, 1) = 0$ if $S = N[D] \cap F^*$ and v has a neighbour in D. Otherwise, we set $A'(T, S, 1) = \infty$. If $d = 2$, this corresponds to the case where $D' = \emptyset$ and v does not have a neighbour in D. We therefore set $A'(T, S, 2) = 0$ if $S = N[D] \cap F^*$ and v does not have a neighbour in D. Otherwise, we set $A'(T', S, 2) = \infty$.

Now consider the case where T contains more than two vertices. Again, it must be of the form $T_1 \leftarrow T_2$, where T_1 and T_2 are trees rooted at v_1 and v_2, respectively, say. We now show how to calculate $A'(T, S, d)$. If $d = 0$, this corresponds to the case where $v_1 \in D'$. Note that $D \cup D'$ must be independent, so $v_2 \notin D'$. Vertex v_1 dominates v_2, so the restriction of $C \cup D'$ to $D \cup T_2$ may or may not dominate v_2. Therefore $A'(T, S, 0)$ is the minimum of $\{A'(T_1, S_1, 0) + A'(T_2, S_2, d') | S_1 \cup S_2 = S, d' \in \{1, 2\}\}$. If $d = 1$, this corresponds to the case where $v_1 \notin D'$, but it is dominated by either D or v_2. Therefore, as for the case of Π^γ, $A'(T, S, 1)$ is the minimum of $\{A'(T_1, S_1, d_1) + A'(T_2, S_2, d_2) | (d_1, d_2) \in \{(1, 0), (1, 1), (2, 0)\}, S_1 \cup S_2 = S\}$. Similarly, $A'(T, S, 2)$ is the minimum of $\{A'(T_1, S_1, 2) + A'(T_2, S_2, 1) | S_1 \cup S_2 = S\}$.

Finally, we define $B'(i, S)$ for the Π^i problem as we did $B(i, S)$ for the Π^γ problem, except that we now demand that $D \cup D'$ is independent. In the forest $V(G) \setminus F^*$, no vertex of any tree can be adjacent to a vertex in different tree. Therefore, the algorithm for calculating $B'(i, S)$ from $A'(T, S, d)$ is identical to that calculating $B(i, S)$ from $A(T, S, d)$ for Π^γ. We complete the proof in the same way as for Π^γ. □

3.2 Intractability results

We now prove the negative results claimed in Table 1.

Proposition 9. *The following hold:*

1. Π^α, Π^γ, Π^χ, Π^i, Π^ν *are* $\notin XP(\chi)$ *and* $\notin XP(\omega)$*;*
2. Π^ω *and* Π^χ *are* $\notin XP(\alpha)$*;*
3. Π^i *is* $\notin XP(\gamma)$*;*
4. $\Pi^\alpha, \Pi^\tau, \Pi^\omega, \Pi^\chi$ *and* Π^ν *are* $\notin XP(\gamma)$ *and* $\notin XP(i)$*.*

Proof. All these results are a consequence of the fact that these problems remain NP-hard if the related parameter is bounded, as shown in the proof of Proposition 3. □

Proposition 10. *The following hold:*

1. Π^ω *is W[1]*(χ)*-hard;*

2. Π^α, Π^τ *and* Π^ν *are W[1]/*(α*)-hard.*

Proof. Claim 1 follows immediately from the W[1]-completeness of MULTI-COLOUR CLIQUE. This problem asks: given a graph G, and a colouring of G with k colours, does G contain a clique on k vertices? Given an instance (G, k) of MULTI-COLOUR CLIQUE, we delete any edges both of whose endpoints are the same colour. The resulting graph G' has $\chi \leqslant k$. Our instance of MULTI-COLOUR CLIQUE is a yes-instance if and only if G' has a clique on k vertices.

For Claim 2, let $G(V, E)$ be a graph and $\overline{G}(V, \overline{E})$ its complement, (i.e. $e \in E \Leftrightarrow e \notin \overline{E}$). Since $\alpha(G) = \omega(\overline{G})$ and $\omega(G) = \alpha(\overline{G})$, the result for Π^α and Π^τ is an immediate consequence of $\Pi^\omega \in$W[1](ω)-hard.

We now prove the Π^ν case. Given a graph $G = (V, E)$, we define G' to be the product of G with a single edge, i.e. $G' = (V', E')$, where $V' = V_1 \cup V_2$ and $E' = \{((v, 1), (v, 2)) | v \in V\} \cup \{((v, i), (u, i)) | i \in \{1, 2\}, uv \in E\}$.

We claim that $\alpha(G') = \alpha(G), \nu(G') = 2(|V| - \alpha(G))$. It is straightforward to verify that $\alpha(G') = \alpha(G)$. Moreover, for any graph G of order n, we have that $\frac{n-\nu(H)}{2} \leqslant \alpha(H)$, so $\nu(G') \geqslant |V(G')| - 2\alpha(G') = 2(|V| - 2\alpha(G))$. On the other hand, for a stable set S of G, $S \times \{1, 2\}$ induces a forest (in fact a matching) and consequently $\nu(G') \leqslant 2(|V| - \alpha(G))$. This completes the proof, since Π^α is W[1](α)-hard. □

Proposition 11. Π^γ *and* Π^i *are W[2]-hard*(α) *and W[2]-hard*(i).

Proof. Since $i \leqslant \alpha$, we need only prove that the problems are W[2]-hard(α). We prove the result for the problem Π^i. Similarly to the case above, given a graph $G = (V, E)$, and $k \in \{1, \ldots, |V|\}$, we define $G_k = (V_k, E_k)$, where $V_k = V_1 \cup \cdots \cup V_k$, $V_i = V \times \{i\}$ and $E_k = E_1 \cup \cdots \cup E_k \cup E'$, where (V_i, E_i) induce cliques and $E' = \{((u, i), (v, j)), i, j \in \{1, \ldots, k\}, i \neq j, v \in N_G[u]\}$.

The following claims then hold:

1. if D_k is an independent dominating set of G_k and $D_k \cap V_j = \emptyset$ for some $j \in \{1, \ldots, k\}$ then $\{v, \exists i, (v, i) \in D_k\}$ is an independent dominating set of G;
2. $\alpha(G_k) \leqslant k$;
3. if $k \geqslant i(G)$ then $i(G_k) = i(G)$.

The first two claims are obvious. In order to prove the third claim, note that if $\{a_1, \ldots, a_{i(G)}\}$ is an independent dominating set in G, then $\{(a_j, j), j = 1, \ldots, i(G)\}$ is an independent dominating set for G_k. Applying the first claim completes the proof.

Thus, given an oracle $\mathcal{O}$ for (Π^i, α), we find a minimum independent dominating set for $G_1, G_2, \ldots$ until such a set has no vertex in some V_i. (We can do this since the constructive and non-constructive versions are equivalent, due to Proposition 6.) The process will finish for $G_k, k \leqslant i(G) + 1$ and since for $j \leqslant i(G) + 1$, we have that $\alpha(G_j) \leqslant j \leqslant i(G) + 1$, we find that $(\Pi^i, i) \leqslant_{FPT} (\Pi^i, \alpha)$. Thus Π^i is indeed W[2]-hard(α). The corresponding result for Π^γ follows similarly. □

4 Conclusion

We have studied the cross-parameterization of MIN VERTEX COVER, MAX INDEPENDENT SET, MAX CLIQUE, MIN COLORING, MIN DOMINATING SET, MIN INDEPENDENT DOMINATING SET and MIN FEEDBACK VERTEX SET. We are aware of the fact that most of the parameters handled in the paper cannot determined in FPT time and that our study is limited to only seven problems and parameters. However, our goal was rather structural than purely algorithmic. We have tried to show that cross-parameterization provides a somewhat deeper insight to the real nature of the parameterized (in)tractability of the problems handled and helps us to better comprehend it.

As one can see in Table 1, any of the problems tackled is FPT with respect to both τ (the standard parameter of MIN VERTEX COVER) and ν (the standard parameter of MIN FEEDBACK VERTEX SET). A natural question to be studied is whether or not there exist natural graph problems that are hard for either τ or ν.

Finally, let us note that cross-parameterization can be applied in many categories of combinatorial optimization problems defined on several structures (and not only on graphs). For instance, what is the parameterized complexity of MIN SET COVER with respect to the standard parameter of MAX SET PACKING or to that of MIN HITTING SET? What are the complexities of the two latter problems with respect to the two remaining parameters?

References

[1] J. Buss and J. Goldsmith. Nondeterminism within p. *SIAM Journal on Computing*, 22(3):560–572, 1993.

[2] M.R. Cerioli, L. Faria, T.O. Ferreira, C.A.J. Martinhon, F. Protti, and B. Reed. Partition into cliques for cubic graphs: Planar case, complexity and approximation. *Discrete Applied Mathematics*, 156(12):2270–2278, 2008.

[3] M. Demange and J. Lorenzo. Approximating values and solutions of NP-optimization problems: concepts and examples. *Foundations of Computing and Decision Sciences*, 26(2):145–168, 2001.

[4] R. Downey and M. Fellows. Fixed-parameter tractability and completeness. *Congressus Numerantium*, 87:161–178, 1992.

[5] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer, New York, 1999.

[6] R.G. Downey and M.R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141(1–2):109–131, 1995.

[7] R.G. Downey and M.R. Fellows. Parameterized computational feasibility. In P. Clote and J. B. Remmel, editors, *Feasible Mathematics II*, volume 13 of *Progress in Computer Science and Applied Logic*, pages 219–244. Birkhäuser Boston, 1995.

[8] J. Flum and M. Grohe. *Parameterized complexity theory.* Texts in Theoretical Computer Science. Springer-Verlag, 2006.

[9] M. R. Garey and D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness.* W. H. Freeman, San Francisco, 1979.

[10] K. Jansen and P. Scheffler. Generalized coloring for tree-like graphs. *Discrete Applied Mathematics*, 75(2):135–155, 1997.

[11] L. Lovász. Coverings and coloring of hypergraphs. *Utilitas Math.*, pages 3–12, 1973. Proceedings of the Fourth Southeastern Conference on Combinatorics, Graph Theory, and Computing.

[12] R. Niedermeier. *Invitation to fixed-parameter algorithms.* Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, Oxford, 2006.

[13] A. Paz and S. Moran. Non deterministic polynomial optimization problems and their approximations. *Theoretical Computer Science*, 15(3):251–277, 1981.

[14] L. Stockmeyer. Planar 3-colorability is polynomial complete. *SIGACT News*, 5(3):19–25, July 1973.